Gp 2431
-TFW

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

Application No.	:	09/725,910	Confirmation No. 9772
Applicants	:	C. Duan	
	:	S. Gupta	
	:	F. Zhao	
Filed	:	November 30, 2000	
Group Art Unit	:	2131	
Examiner	:	M. Henning	
Docket No.	:	3731-0141P	
Title	:	VARIABLE SIZE KEY CIPHER AND METHOD AND DEVICE USING THE SAME	

Commissioner for Patents
PO Box 1450
Alexandria, Virginia 22313-1450

TRANSMITTAL OF CERTIFIED COPY OF PRIORITY DOCUMENT

Sir:

Enclosed for filing in the above referenced patent application is a certified copy of the following priority document:

Chinese Patent Application No.: 00134266.5
Filed: November 27, 2000

Applicants' priority claim under 35 U.S.C. § 119 is hereby perfected.

Respectfully submitted,

Jeffrey M. Weinick
Reg. No. 36,304
Attorney for Applicants
Tel.: 973-533-1616

Date: August 31, 2004
Law Office of Jeffrey M. Weinick, LLC
615 West Mount Pleasant Avenue
Livingston, NJ 07039

I hereby certify that this correspondence, as well as any items referred to as being transmitted herewith, is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on August 31, 2004.

Typed or printed name of person signing this certificate:

Risa Garcia

Signature: Risa Garcia

证　　明

本证明之附件是向本局提交的下列专利申请副本

申　　日：　2000.11.29

申　　号：　00134266.5

申　　别：　发明

发明创造名称：　可变大小的密钥以及使用该密钥的方法和装置

申　　人：　朗迅科技公司

发明人或设计人：　赵风光、段成罡、苏尼尔·K ·古普塔

中华人民共和国
国家知识产权局局长　王景川

2004 年 8 月 3 日

权 利 要 求 书

1. 一种加密装置，包含：

一个随机数生成器，接收主密钥，使用至少一个随机数确定工作密钥，并输出该工作密钥；

一个模型，接收主密钥、工作密钥和明文，并产生至少两个频率计数；以及

一个编码器，根据工作密钥、明文以及至少两个频率计数输出密文。

2. 根据权利要求书 1 的加密装置，其中由所述随机数生成器产生的工作密钥的长度是可变的。

3. 根据权利要求书 1 的加密装置，其中所述编码器的输出是可变的。

4. 根据权利要求书 1 的加密装置，其中工作密钥和主密钥不相同。

5. 根据权利要求书 1 的加密装置，其中所述模型包含至少一个频率表，该频率表包含至少两个频率计数。

6. 根据权利要求书 1 的加密装置，其中至少一个频率表存储在一个 RAM 中。

7. 根据权利要求书 1 的加密装置，其中由所述编码器输出的加密文本是根据基于比特的处理方案。

8. 根据权利要求书 5 的加密装置，其中至少一个频率表包含工作密钥。

9. 一种加密方法，包含：

处理随机比特和密钥比特以产生至少一个频率表；以及

使用至少一个频率表编码明文。

10. 根据权利要求书 9 的方法，其中所述处理步骤包含产生一个长度等于密钥的随机比特字符串。

11. 根据权利要求书 9 的方法，其中的所述处理步骤中，不同的密钥比特产生至少一个不同的频率表。

12. 根据权利要求书 9 的方法，其中所述编码步骤的输出是可变的。

13. 根据权利要求书 9 的方法，其中由所述编码器输出的加密文本是根据基于比特的处理方案。

14. 根据权利要求书 11 的方法，其中至少一个频率表包含工作密钥。

15. 一种解密装置，包含:

一个模型，接收主密钥、工作密钥和明文，并产生至少两个频率计数;

一个解码器，根据工作密钥、主密钥、明文、至少两个频率计数输出明文，以及

一个随机数生成器，接收明文，并利用至少一个随机数确定工作密钥，以及输出工作密钥到所述模型。

16. 根据权利要求书 15 的解密装置，其中由所述随机数生成器产生的工作密钥的长度是可变的。

17. 根据权利要求书 15 的解密装置，其中所述解码器的输出是可变的。

18. 根据权利要求书 15 的解密装置，其中工作密钥和主密钥不相同。

19. 根据权利要求书 15 的解密装置，其中所述模型包含至少一个频率表，该频率表包含至少两个频率计数。

20. 根据权利要求书 19 的解密装置，其中至少一个频率表存储在一个 RAM 中。

21. 根据权利要求书 15 的解密装置，其中由所述编码器输出的加密文本是根据基于比特的处理方案。

22. 根据权利要求书 19 的解密装置，其中至少一个频率表包含工作密钥。

23. 一种解密方法，包含:

处理随机比特和密钥比特，以产生至少一个频率表; 以及

采用至少一个频率表解码密文。

24. 根据权利要求书 23 的方法，其中所述处理步骤包含产生长度等于密钥的一个随机比特字符串。

25. 根据权利要求书 23 的方法，其中所述的处理步骤中，不同的密钥比特产生至少一个不同的频率表。

26. 根据权利要求书 23 的方法，其中所述解码步骤的输出是可变的。

说　　明　　书

可变大小的密钥以及使用该密钥的方法和装置

本发明针对加密和解密，尤其是针对一种可变大小的密钥，以及利用可变大小的密钥来执行加密和解密方法和装置。

传统地，压缩和密码学被认为是界限清晰和独立的技术，被分别开发和应用。然而，尽管它们是以各自不同的方式来工作的，但它们具有去除输出冗余的共同目标。认识到这一共同目标，Witten，Neal 和 Cleary（下文中称为 WNC）首次应用自适应算术编码来加密。特别的，WNC 作出了如下发现：

· 通过再编码消息，压缩保护了消息不会被偶然发现；

· 去除冗余，摒弃了密码分析学家利用自然语言中的通常统计规律来分析的可能性；以及

· 自适应地利用被传输数据的特征，可提供较好的压缩性能。

在这三个发现中确定的特性，显然给出了良好的压缩性能和安全性能——两方面都很完美。

图 1 示意了基于模型的、常规、通用和算术编码的加密方案 10，诸如 WNC 方案的原理流程图。如图 1 所示，明文 12 被输入到编码器 14 和模型 16，密钥 18 也被输入到模型 16。编码器 14 根据明文 12 以及模型 16 的输出产生密文 20。模型 16 在任何给定上下文中，为下一个字符提供概率分布。最简单的模型对上下文不敏感，而不管相邻的字符如何都给予同样的分布。模型 16 不应该对实际出现的字符指定 0 分布，否则该符号就不能被编码，因为符号范围的顶端和底端相重合。对于编码器 14，选择一个源符号字母，而且为每个符号指定一个出现概率。符号范围间隔通常为 0 到 1，而且每个源符号根据其概率占据该间隔中的一个子间隔。随着每个新的源符号的读取，该间隔被连续细分。高概率符号减小的间隔量要少于低概率符号的减小量。密文 20 由间隔中的一个值表示，这种系统在计算机和安全出版社出版的“Data Security in a Fixed-Model Arithmetic Coding Compression

Algorithm", 11(1992), pp.445-461 中有描述。

如同算术编码这个名称所提示的，组成明文 12 的源符号根据数字编码。每个符号在其每次被编码时，不必都翻译成同一固定码来组成密文 20。可能是一个源符号字符串的输入字符串，通常由 0 和 1 之间的真实数字的一个间隔来表示。间隔的范围最初可由与讨论中的符号的概率成正比的一个值来定义。随着每个新的源符号从明文 12 中读取，间隔可被连续细分，明文 12 中的高概率符号的间隔减小量要少于低概率符号的减小量。作为模拟，图 1 所示的算术编码类似于使用一种灵活的标尺来测量符号字符串。

WNC 方案是一种用于加密的基于字节的算术编码方案，它使用频率表，但不用随机数生成器。WNC 方案的密钥特征为，一个基于字节的模型以及一个初始频率表用作加密的密钥。而在 WNC 中，工作密钥和主密钥是相同的。

然而，其后 Bergen 等人在计算机和安全出版社出版的"Data Security in a Fixed-Model Arithmetic Coding Compression algorithm", pp.445-461,1992 一文中的研究表明，WNC 方案存在安全问题。特别是，作为固定模型算术编码方案的 WNC 实现容易被分析，因此可能被简单和直接解密。这种容易被分析和解密是由于在输出中重复使用固定子字符串的直接结果，这种固定子字符串能特征化每个特定的符号。WNC 实现的固定特征使得，能相对简单地确定原始频率表中的符号顺序以及符号频率的实际值。因此很难为 WNC 加密方案设计一种安全模型和密钥控制。

本发明通过提供一种加密装置和方法以及一种解密装置和方法，采用传统的算术编码技术来解决上述问题，该编码技术根据的是一种基于比特的算术编码技术。该加密装置和方法以及该解密装置和方法使用值 0 和 1 的频率表以及一个随机数生成器。频率表包含工作密钥而非主密钥，均采用传统技术。在开始编码时，主密钥被输入到一个编码器。一个模型使用根据主密钥的频率表以及一个随机比特来形成一个工作密钥。工作密钥为可变的，其用作编码明文的概率。本发明中的模型根据输入文本来刷新概率。

具体来讲，本发明针对一种加密装置，它包含一个随机数生成器，用于接收主密钥，利用至少一个随机数来确定一个工作密钥并输出该工作密钥；一个模型，用于接收主密钥、工作密钥和明文并产生至少两个频率计数；以及一个编码器，基于工作密钥、明文以及至少两个频率计数来输出密文。

此外，本发明针对一种加密装置，它包含处理随机比特和密钥比特以产生至少一个频率表；以及利用至少一个频率表来编码明文。另外，本发明还针对一种解码设备，它包含一个模型，用于接收主密钥、工作密钥以及明文，并产生至少两个频率计数；一个解码器，基于工作密钥、主密钥、明文、至少两个频率计数来输出明文；以及一个随机数生成器，用于接收明文，并利用至少一个随机数来确定工作密钥，以及输出该工作密钥到所述模型。此外，本发明还针对一种解密方法，它包含处理随机比特和密钥比特以产生至少一个频率表；以及利用至少一个频率表来解码密文。

本发明的基于比特的加密方案和硬件设计能产生一种密码，该密码基于流结构，且具有长度不受限制的密钥。该密码还有一个优点是，可压缩明文至少 50%。对于相同的明文和相同的密钥，该密码可随不同的环境而变化。在硬件设计中的操作是基于算术加法和位移指令，而不包含乘法和除法指令。因此，硬件设计更简单。密码、编码器、解码器以及方法也适用于密码学和电子商务。

图 1 示意了一种基于模型的常规、通用、算术编码的加密方案。

图 2 示意了在本发明的一个示例性实施例中，加密的示例性原理流程图。

图 3 示意了在本发明的一个实施例中，编码的流程图；以及

图 4 详细示意了在本发明的一个示例性实施例中的模型。

图 5 示意了在本发明的一个示例性实施例中的频率表。

图 6 示意了在本发明的一个示例性实施例中，解密的示例性原理流程图。

图 7 示意了在本发明的一个实施例中的解码流程图；以及

图 8 详细示意了在本发明的另一个示例性实施例中的模型。

图 2 示意了在本发明的一个示例性实施例中，加密的示例性原理流程图。如图 2 所示，明文 12 被输入到编码器 114 和模型 116。主密钥 118 提供给模型 116 和随机数生成器 122。随机数生成器 122 根据主密钥 118 以及其中生成的随机数产生工作密钥，工作密钥被输出到编码器 114 和模型 116。模型 116 将 0 和 1 的两个频率计数分别提供给编码器 114，而且编码器 114 的输出为一个压缩比特流。编码器 114 根据明文 12、随机数生成器 122 输出的工作密钥以及来自模型 116 的频率计数 0、1，来产生压缩信息（即，密文 120）。

编码器 114 可如下述操作。要编码的消息由 0 和 1 之间的真实数字的间隔表示。随着消息长度的增加，需要用来表示消息的间隔减小，而需要用来指定间隔的比特数增加。消息中的后续符号根据模型 116 产生的符号概率来减小间隔的大小。高概率符号减小的范围越少于低概率符号的减小范围，因此增加到消息中的比特数越少。

最初，指定到一条消息的间隔为整个间隔[0,1)（[0,1)表示半开间隔 $0 \le x < 1$]。随着消息中的每个符号被处理，范围被限制到分配给指定符号的范围部分。例如，假定字母为（a,b,c,d,e,f），使用一种固定模型，其概率如表 1 所示。

表 1

符号	概率	范围
a	0.25	[0,0.25)
b	0.25	[0.25,0.5)
c	0.1	[0.5,0.6)
d	0.1	[0.6,0.7)
e	0.1	[0.7,0.8)
f	0.2	[0.8,1.0)

假定发送的消息为 abc。起初，编码器 114（以及后面将要描述的一个相关的解码器）知道该范围为[0,1)。接收到第一个符号 a 之后，编码器 114 就将范围缩小至[0,0.25)，即模型 116 分配给符号 a 的范围。

第二个符号 b 将新的范围再次缩至 1/4，即[0.0625,0.125)——前一范围为 0.25 个单位长度，它的 1/4 为 0.0625。下一个符号 c 被分配[0.5,0.6)，当此范围应用到[0.0625,0.125)时，提供更窄的范围[0.09375,0.1)。

假定所有的相关解码器了解到该消息的最终范围为[0.09375,0.1)。解码器就能立即推断出第一个字符为 a，因为其范围完全位于表 1 模型分配给 a 的空间内。这之后，范围为[0,0.25)。在看到完全包含给定范围[0.09375,0.1)的 b 范围[0.0625,0.125) 之后，就确定第二个字符为 b。如此继续，解码器就能识别出所有消息。

在一个示例性实施例中，编码器 114 为同时待审的美国专利申请号 No.09/240,576 中所描述的编码器，该申请名为“Multiplication-Free Arithmetic Coding”,1999 年 2 月 1 日申请，该申请的全部内容在此作为参考。这种编码器的一个优点是，其中不涉及乘法和除法操作，这使得硬件设计很简单。这个编码器将在下面描述。

编码

最初，两个寄存器 R 和 L 分别设置为 1 和一个任意数。编码器 114 提供有三个输入，第一个频率计数 c_0 代表概率 0 的分数值，第二个频率计数 c_1 代表概率 1 的分数值，以及在此范围内编码的一个符号 i（0 或 1）。

编码器 114 执行的编码步骤可归述为下面的伪代码:

1. If $c_0>c_1$, exchange the values of c_0 and c_1, and let i=!i.
2. While $R\leq c_1$, do
- Output the most significant bit of L.
- L = L<<=1, R = R <<=1.
- If R>~L, then R = ~L+1.
3. If i=0, then $R=c_0$; else $R = R-c_0$, $L = L+c_0$.

输出 L

应注意的是，在上面的伪代码中使用了一些 C 语言符号。！代表逻辑补（*logic complement*），~ 代表二进制补（*binary complement*），而 <<= 代表算术左移位（*arithmetic shift left*）。根据上面的描述，本发明基于下面的假设而工作：对于每个迭代，$R \approx c_0 + c_1$。

$L := L,\ R := c_0,\ i = 0$ （1）

$L := L + c_0,\ R := R - c_0,\ i = 1$ （2）

在本发明中，将两个寄存器R和L分别初始化为1和一个任意数，使输出流中的第一个字来指示一个同步字，用于实时传输。此外，步骤1通常称为交换步骤，步骤2称为调整步骤，步骤3称为编码步骤。常规的无乘法算术编码技术中要求使用的量化（magnitude）步骤，在本发明中不要求。在本发明中，调整步骤先于编码步骤执行。在调整步骤中，当寄存器R的值小于或等于第二个频率计数的值时，执行“while”循环，如果寄存器R的值大于寄存器R值的二进制补时，将寄存器R的值设置为等于寄存器L值的二进制补加1，由此就不需要下一比特的塞入步骤。

总之，本发明的无乘法算术编码方法通过下述步骤来产生一个编码比特流：接收来自一个编码字符串的一个符号和两个频率计数，发现一个最大概率和最小概率符号；提供第一寄存器用于幅度位移操作，以输出比特到编码的比特流，为编码字符串中的每个符号近似一个上下文概率，以及根据上下文概率来编码编码字符串中的下一个符号。

图3包含了编码器114在编码进程20中执行的详细的特定步骤。特别的，在步骤22，寄存器R和L分别被初始化为1和同步字。在本例中，编码比特流为11011100i，它与寄存器R和L的初始值在步骤24一起输入到0阶马尔可夫模型，以产生频率计数c_0和c_1。在步骤26，比较c_0和c_1，如果c_0大于c_1，那么c_0和c_1互换，而且在步骤28，i被设置为其逻辑补。然而，如果c_0不大于c_1，进程继续到步骤30，在此确定寄存器R的值是否大于或等于c_1。如果是这样，进程继续到步骤32，在此输出寄存器L的最高有效位（MSB），L和R被算术左移位，而且如果R大于L的二进制补，那么R被设置为L的二进制补加1，而且进程继续到步骤30。如果寄存器R的值不大于等于C_1，那么进程继续到步骤34。在步骤34，确定i是否等于0。如果i等于0，那么寄存器R的值在步骤36被设置为等于C_0，如果i不等于0，那么在步骤38，R被设置为R的前一值减c_0，而且L被设置为L的前一值加c_0，从而编码比特流中的下一个比特。该进程继续执行，直到输入比特流

中的所有比特被编码。接着，输出寄存器 L 的值作为编码的比特流。

尽管本发明是采用 0 阶马尔可夫模型描述的，但均可使用本领域的普通技术人员所了解的任何模型。

如图 4 所示，模型 116 包含频率表 130（示意为 RAMs126）以及模型控制器 128。包含在频率表 130 中的频率计数代表概率，如表 1 中所示的概率。明文 12、主密钥 118 和工作密钥被输入到模型控制器 128。随机数生成器 122 每个系统时钟产生一个随机比特。如图 4 所示，频率表 130 可能包含两个相关项，因此使得很难跟踪保存在频率表 130 中的所有信息，除了两个相关项。模型 116 可利用地址寄存器 r 来记录当前处理的最近 t 比特，频率表 130 的大小为 2^t。在一个实施例中，模型 116 为 t 阶马尔可夫模型，而 r 看起来像大小为 t 的滑动窗口。最初，频率表 130 中的值可设置为 1。

本发明可描述为两阶段密码，第一阶段处理随机比特和密钥比特。在第一阶段，密钥大小控制随机比特生成器，以便控制器 128 能得到与密钥相同大小的随机比特字符串。对于每个比特对（一个随机比特和一个密钥比特），控制器 128 可执行下述操作：

1） 根据模型控制器 128 中的一个移位寄存器，从 RAMs 126 得到 F_0 和 F_1；

2） 如果密钥比特为 0，F_0 加 1；否则 F_1 加 1；

3） 传递随机比特和 F_0、F_1 到编码器 114；

4） 如果随机比特为 0，F_0 加 1；否则 F_1 加 1；

5） 回写 F_0 和 F_1 到 RAMs 126；

6） 左移位模型控制器 128 中的移位寄存器，并将当前随机比特插入到移位寄存器中的最末位。

在第一阶段，随机比特经模型控制器 128 提供给编码器 114（或解码器）。当第一阶段完成后，可在 RAMs 126 获得一个有用的初始频率表。

在第二阶段，编码明文 12。第二阶段中，明文 12 被输入到模型控制器 128，模型控制器 128 为每个输入比特执行如下动作：

1） 根据移位寄存器，从 RAMs 126 得到 F_0 和 F_1；

2） 传递明文比特和 F_0、F_1到编码器 114;

3） 如果明文比特为 0，F_0加 1；否则 F_1加 1;

4） 回写 F_0和 F_1到 RAMs 126;

5） 左移移位寄存器，并将当前明文比特插入到移位寄存器的最末位。

由此，明文 12 也会经模型控制器 128 传递到编码器 114（或解码器）。

图 5 示意了本发明的一个优选实施例中的频率表 130。如图 5 所示，频率表 130 包含 r 个频率 0 输入和 r 个频率 1 输入。频率表 130 的大小在一个实施例中为 2^t。在一个实施例中 t = 15。

模型控制器 128 控制 RAMs 126 的读写、频率表 130 的输出以及到算术编码器 114 的源比特。到编码器 114 的输入包含来自明文 12 的文本比特、来自主密钥 118 的密钥比特、来自随机数生成器 112 的随机比特，以及来自 RAMs 126 的两个频率 136。模型控制器 128 到 RAMs 126 的输出为使读（read-enable）信号 138、使写（write-enable）信号 140、比特“0”和“1”的各自修改频率 142，以及地址 144。从模型控制 128 到编码器 114 的输出包含源比特 146，以及比特“0”和“1”的一对频率计数 148。在一个示例性实施例中，模型 116 使用两个时钟实现，一个系统时钟和一个 RAM 时钟，目的是使模型控制器 128 能在一个系统周期内完成到 RAMs 126 的一次读和写。

编码器 114 和模型 116 之间的相互作用如下所述：最初，r 可设置为一个固定数，r 的当前值用于从频率表 130 中查找 0 和 1 的各自频率计数。这两个频率计数接着被输入到编码器 114。在由 r 指向的位置，当前比特被编码，频率计数被刷新。接下来，滑动 r 以包含当前比特，重复此操作直到编码完毕所有比特。

如图 4 中的实施例所示，频率表 130 包含随机存取存储器（RAM）126。两个 RAMs 126 分别表示比特 0 和 1 的频率表。在一个示例性实施例中，总共有 64k 对比特“0”和“1”的频率。因此，频率可以在从 1 到 255 的范围内。编码器 114 实现一种算术编码算法，在此，其输入信号为一个比特源信号和比特“0”和“1”的一对频率。对于每

个时间间隔，该频率对都不相同，且取决于输入源比特。编码器 114 的输出为密文 120 和一个输出有效比特 150。

本发明也可使用一个密钥（任何长度的比特字符串）来控制频率表 130 中的初始值，以及使用一个随机比特流来控制 r 的值。随机比特流可由随机数生成器 122 产生。用于加密的密钥被称为工作密钥。更精确地说，k_1, k_1,..., k_n 为加密密钥的比特流。一个示例性算法如下所述：

加密

初始化：r = 0，使频率表 130 中的所有项为 1，初始化编码器 114，j = 1

输入：k_1, k_1,..., k_n

1. while j <= n, do

- 从频率表 130 中查找由 r 指定的位置。
- 如果 $k_j = 1$，频率 1 位置加 1，否则频率 0 位置加 1。
- 采用当前的频率计数来编码来自随机数生成器 122 的一个比特 *l*。
- 如果 *l* = 1，频率 1 位置加 1，否则频率 0 位置加 1。
- 左移位 r，r = r | 随机比特。

2. 编码明文 12 和刷新模型 116，如下：

- 如果当前比特为 1，频率 1 位置加 1，否则频率 0 位置加 1。
- 左移位 r，r = r | 当前比特。

应注意的是，步骤 1 用于产生初始频率表 130，因为采用随机数生成器 122，频率表 130 可取决于环境。此外，即使不同时候采用同一加密密钥，也将产生不同的频率表 130，这表明本发明中的密码不是一对一，而是可变的。

在一个优选实施例中，VHDL 语言用于描述图 4 示意的模型控制器 128 和编码器 114 之间的行为模式。示例性的 VHDL 提供如下：

```
library IEEE;
-- use IEEE.std_logic_unsigned.all;
 use IEEE.std_logic_signed.all;
 use IEEE.std_logic_arith.all;
 use IEEE.std_logic_1164.all;

entity cipher is
 port(
  key     : in std_logic;
  random  : in std_logic;
  text    : in std_logic;
  end_of_key : in std_logic;
  end_of_text: in std_logic;
  data0_in  : in std_logic_vector(7 downto 0);
  data1_in  : in std_logic_vector(7 downto 0);
  sys_clock  : in std_logic;
  mem_clock  : in std_logic;

  data0_out  : out std_logic_vector(7 downto 0);
  data1_out  : out std_logic_vector(7 downto 0);
  addr     : buffer std_logic_vector(15 downto 0);
  READ_ENABLE : out std_logic;
  WRITE_ENABLE: out std_logic;


  cipher_text : out std_logic;
  out_valid  : out std_logic
 );
end cipher;

architecture RTL of cipher is

signal encode_bit   : std_logic;
signal encode_valid  : std_logic := '0';
signal c0,c1       : std_logic_vector(7 downto 0);
signal L          : std_logic_vector(31 downto 0) := "00000000000000000000000000000000";
signal R,H        : std_logic_vector(31 downto 0) := "00000000000000000000000000000001";
signal freq0       : std_logic_vector(7 downto 0);
signal freq1       : std_logic_vector(7 downto 0);
signal j          : std_logic;

begin

model_update : process
 variable a,b  : integer;
 variable counter : integer := -1;

 begin
  wait until mem_clock'event and mem_clock='1';
  counter := (counter + 1) mod 8;

  case counter is
   when 0 => READ_ENABLE <= '1';
         WRITE_ENABLE <= '0';
         if(end_of_key='0') then
          addr <= SHL(addr,"01") + key;
         else
          addr <= SHL(addr,"01") + text;
         end if;

 when 2 => if(end_of_key='0') then
         encode_bit <= random;
         c0 <= data0_in + not key;
         c1 <= data1_in + key;
         a := conv_integer(data0_in) + conv_integer(not key) + conv_integer(not random);
         b := conv_integer(data1_in) + conv_integer(key) + conv_integer(random);
        else
         encode_bit <= text;
         c0 <= data0_in;
         c1 <= data1_in;
         a := conv_integer(data0_in) + conv_integer(not text);
         b := conv_integer(data1_in) + conv_integer(text);
        end if;
```

```
        if(end_of_text='0') then
         encode_valid <= '1';
        else
         encode_valid <= '0';
        end if;

        READ_ENABLE <= '0';
        WRITE_ENABLE <= '1';
        if(a>16#FF# or b>16#FF#) then
         data0_out <= shr(conv_std_logic_vector((a+1),8),"01");
         data1_out <= shr(conv_std_logic_vector((b+1),8),"01");
        else
         data0_out <= conv_std_logic_vector(a,8);


          data1_out <= conv_std_logic_vector(b,8);
        end if;
    when others => null;
   end case;
end process;

arith_coder : process
 variable a,b,c : std_logic_vector (31 downto 0);
 variable f0,f1 : std_logic_vector (7 downto 0);
 variable k   : std_logic;

 begin

  wait until sys_clock='1' and sys_clock'event;
  if(encode_valid='1') then
   if(c0>c1) then
    freq0 <= c1;
    freq1 <= c0;
    j <= not encode_bit;
   else
    freq0 <= c0;
    freq1 <= c1;
    j <= encode_bit;
   end if;

   f0 := freq0;
   f1 := freq1;
   k := j;

   while R<=f1 loop
    cipher_text <= L(31);
    out_valid <= '1';

    a := shl(L,"01");
    b := shl(R,"01");
    c := not a;

  L <= a;
  if b>a then
   R <= c + '1';
  else
   R <= b;
  end if;

  wait until mem_clock='1' and mem_clock'event;
 end loop;

 out_valid <= '0';
 if(k='0') then
  R <= conv_std_logic_vector (conv_integer(f0),32);
 else
  R <= R - f0;
  L <= L + f0;
 end if;

 end if;

end process;

end RTL;
```

实例

本例中用于测试的参数如下：

L---编码间隔的低端：32 bits，初始化为 0；

H---编码间隔的高端：32 bits，初始化为 1；

R---编码间隔的范围：32 bits，初始化为 1；

V---用于解码比特流的寄存器；

2^t---频率表 130 的大小：对频率 0 和 1 均为 64k，t = 15；

r---表格的地址指针寄存器：15 bits。

表 2：明文“AAAAAAAAAAAAAAAAAAAAAAAA”

实验号	密钥	密文（HEX）
1	A	B9 50 C8 C9 1B F8 44 10
2	A	FA A1 91 91 3C 81 14 80
3	A	83 D3 C3 C7 28 1F 35
4	AbCD8910	56 DF B4 56 89 48 67 9E 82 28 28 28 66 45 21 40
5	AbCD8910	B9 72 D9 5D A0 F1 62 68 99 7D 7D 70 98 EE F8

表 3：明文“It is incredible for us”

实验号	密钥	密文（HEX）
1	Zfg	E7 95 CE 8C A3 B7 7E 1D 98 9E 1E 6F 0D 77 32 14 C5 58 24 4B FF 40 69 43 1C 45 29 80
2	Zfg	2B CF 08 FD 5F 54 87 E1 D9 89 E1 E6 F0 D7 73 21 4C 55 82 44 BF F4 06 94 31 C4 52 98
3	123	C4 99 9E F4 97 49 27 32 06 97 32 0A 0B 62 86 25 13 CA 51 2E 44 BA 86 72 45 CA 95 27 00
4	123	E5 3D C0 5C 82 58 12 EA 84 95 52 85 69 0D 77 32 14 C5 58 24 4B FF 40 69 43 1C 45 29 80

从上面的表 2 和表 3 显然可见：1）对于具有相同密钥的相同明文，可产生不同的密文，2）密文的大小可随不同的实验参数和不同密钥而改变，以及 3）对于高度相关数据，压缩率很高，但对于弱相关数据或较短的字符串，压缩率也还不错。

如果频率表中的值用作加密密钥，本发明的技术也可用于加密。本发明与 WNC 之间的一个差别在于模型。本发明的基于比特的模型，使得利用诸如 Bergen/Hogen 所描述的技术很难跟踪所有初始值。压缩的比特流或密文 120 可通过反向处理进程来解码。

图 6 示意了在本发明的一个实施例中，解密的示例性原理流程图。如图 6 所示，密文 120 输入到解码器 124。主密钥 118 被输入到模型 116 和解码器 124。随机比特生成器 152 的输出被输入到模型 116。模型 116 的输出被输入到解码器 124。解码器 124 解码密文 120 以产生明文 12，明文 12 又反馈回模型 116。解码器 124 也将其输出传递到随机比特生成器 152。在一个示例性实施例中，解码器 124 为在同时待审的美国专利申请序列号 No.09/240,576 中描述的解码器，名称为“Multiplication-Free Arithmetic Coding”，1999 年 2 月 1 日申请，该申请的全部内容在此作为参考。这种解码器将在下面详细描述。

解码

为了解码，R 和 L 寄存器再次被初始化，使用第三个寄存器 V 来存储部分解码比特流，而 i 表示输出比特。如果 S 为解码比特流，它由上述的编码算法产生，解码器 124 执行的解码步骤可归述为下面的伪代码:

1. If $c_0>c_1$, exchange the values of c_0 and c_1, and let i = 1; else i = 0。

2. While $R\leq c_1$, do

- L = L<<=1, R = R <<=1, V =V <<=1。
- V = V | next bit from S.
- If R>~L, then R = ~L+1.

3. If $c_0<V$, then $R = c_0$; else $R = R-c_0$, $L = L+c_0$, and i = !i

总之，用于产生一个解码字符串的无乘法算术编码方法，接收来自解码流的比特和两个频率计数，查找一个最高概率符号和一个最低概率符号，将第一寄存器提供用于幅度位移操作，以从解码比特流中

输入比特，近似解码字符串中每个符号的上下文概率，以及根据上下文概率来解码下一个字符到解码流。

图 7 包含解码器 124 在解码进程 40 中执行的详细的特定步骤，尤其是，在步骤 42，寄存器 R、L 和 V 被初始化。寄存器 R、L 和 V 的值以及要被解码的字符串在步骤 44 被输入到 0 阶-马尔可夫模型，以产生频率计数 c_0 和 c_1。在步骤 46，比较 c_0 和 c_1，如果 c_0 大于 c_1，那么 c_0 和 c_1 互换，而且在步骤 48，i 被设置为其逻辑补。然而，如果 c_0 不大于 c_1，进程继续到步骤 50，在此确定寄存器 R 的值是否大于等于 c_1。如果是这样，进程继续到步骤 52，在此寄存器 R、L 和 V 都被算术左移位，来自解码比特流 S 的下一个比特被添加到寄存器 V，而且如果 R 大于 L 的二进制补，那么 R 被设置为 L 的二进制补加 1。进程接着返回到步骤 50.

如果寄存器 R 的值不大于等于 c_1，那么进程继续到步骤 54。在步骤 54，确定 c_0 是否小于 V。如果 c_0 小于 V，那么寄存器 R 的值在步骤 56 被设为等于 c_0，如果 c_0 不小于 V，那么在步骤 58，R 被设置为 R 的前一值减 c_0，L 被设置为 L 的前一值加 c_0，而 i 被设置其逻辑补，从而解码比特流 S 中的下一个比特。进程接着通过输入下一个比特到步骤 44 的马尔可夫模型刷新而重复执行。进程继续进行，直到解码比特流 S 中的所有比特被解码。

尽管本发明刚刚使用 0 阶 Markov 模型描述，但均可使用本领域的普通技术人员所了解的任何模型。

下面的表 4 示意了对于不同的文件类型来说，采用乘法的编码器、美国专利 No.4,652,856 中公开的现有技术，以及本发明的无乘法算术编码之间的压缩率比较。

表 4

源文件	文件大小	乘法编码器	本发明的编码器	美国专利 No.4,652,856
C 源文件	27620	37.5%	38.4%	39.9%
中文文件	72596	43.3%	43.8%	44.9%
比例图像	262330	67.9%	68.8%	69.6%
EXE 文件	54645	74.3%	74.6%	75.6%
混合数据	417192	67.2%	68.0%	68.9%

如表 4 所示，本发明实现了一种优于现有技术的无乘法算术技术的压缩比。表 4 也示意了乘法编码器通常能提供最佳的压缩，因为每个无乘法设计使用一些近似值而不是实际概率，所以使用无乘法算术技术经常会降低压缩比。然而，表 4 所示的本发明提供了更低的计算难度和低成本的硬件实现，而且仍可实现能与基于乘法的技术相比拟的压缩比。

如图 8 所示，主密钥 118 提供给模型控制器 128。模型控制器 128 控制 RAMs 126 的读写，以及频率表 130 的输出和到解码器 124 的源比特。解码器 124 的输出包含来自密文 120 的文本比特，来自主密钥 118 的密钥比特，以及比特“0”和“1”的一对频率计数 148。模型控制器 128 到 RAMs 126 的输出为使读信号 138、使写信号 140、比特“0”和“1”的各自修改频率 142，以及地址 144。RAMs 126 输出两个频率 136 到模型控制器 128。在一个示例性实施例中，模型 116 使用两个时钟实现，即系统时钟和 RAM 时钟，目的是使模型控制器 128 在一个系统周期内完成到 RAMs 126 的读和写。

本发明也可描述为两阶段的解密。在第一阶段，随机比特从密文比特中被解码。在第一阶段，密钥的大小控制解码器 124，以便模型控制器 128 能从解码器 124 接收与密钥同样大小的随机比特流。对于每个比特对（一个随机比特一个密钥比特），解密执行如下：

1） 采用解码器 124 中的一个移位寄存器，从 RAMs 126 得到 F_0 和 F_1；

2） 如果密钥比特为 0，F_0加 1；否则 F_1加 1；

3） 传递 F_0，F_1到解码器 124；

4） 解码器 124 解码随机比特，并将随机比特送至模型控制器 128；

5） 如果随机比特为 0，模型控制器 128 加 1 到 F_0；否则加 1 到 F_1；

6） 回写 F_0和 F_1到 RAMs 126，以及

7） 左移位寄存器，并将当前随机比特插入到移位寄存器的最末位。

第一阶段完成后，可在 RAMs 126 中获得一个有用的初始频率表。

在第二阶段，解码明文 12。第二阶段中，只要求一个输入，即密文 120，而且对于每个输入比特，解密包含下述步骤：

1） 根据移位寄存器，从 RAMs 126 得到 F_0和 F_1；

2） 传递 F_0，F_1到解码器 124；

3） 解码器 124 解码明文比特，并将明文比特送至模型控制器 128；

4） 如果明文比特为 0，F_0加 1；否则 F_1加 1；

5） 回写 F_0和 F_1到 RAMs 126，以及

6） 左移寄存器，并将当前明文比特插入到移位寄存器的最末位。

因此明文 12 将从解码器 124 中输出。

为解码加密的消息，可建立频率表 130，而且明文 120 中的随机比特流可在解码开始之前恢复。解码也可以下述伪代码定义：

解密

1. while j <= n, do

- 从频率表 130 中查找由 r 指定的位置。
- 如果 $k_j = 1$，频率 1 位置加 1，否则频率 0 位置加 1。
- 采用当前的频率计数来解码一个随机比特 *l*。
- 如果 $l = 1$，频率 1 位置加 1，否则频率 0 位置加 1。
- 左移位 r，r = r | 随机比特。

2. 解码密文 120 和刷新模型，如下：

- 如果当前比特为 1，频率 1 位置加 1，否则频率 0 位置加 1。
- 左移位 r，r = r | 当前比特。

应注意的是，图 1-3，6 和 8 中的功能块可以硬件和/或软件实现。硬件/软件实现可包含处理器和产品的组合。产品可进一步包含存储介质和计算机可执行程序。计算机可执行程序可包含执行所述操作的指令。计算机可执行程序也可提供作为外部提供的传播信号的一部分。

上面已经描述了本发明，显然，本发明可通过许多方式改进。这些改进不认为是偏离本发明的精神和范围，而且，所有这些改进显然对本发明的技术人员来说，是包含在所附的权利要求书的范围之内。

说 明 书 附 图

图1

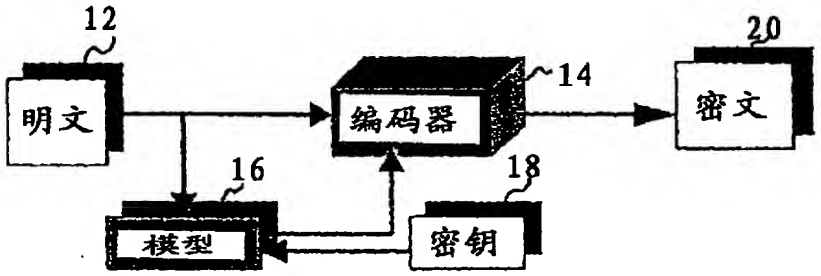

12
明文
编码器
14
20
密文
16
模型
18
密钥

图2

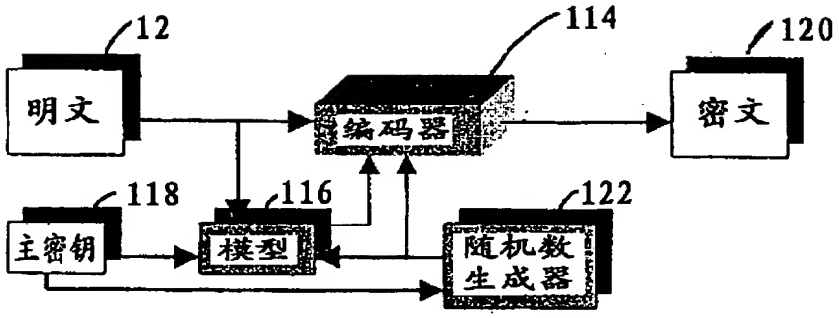

12
明文
114
编码器
120
密文
118
主密钥
116
模型
122
随机数
生成器

图 3

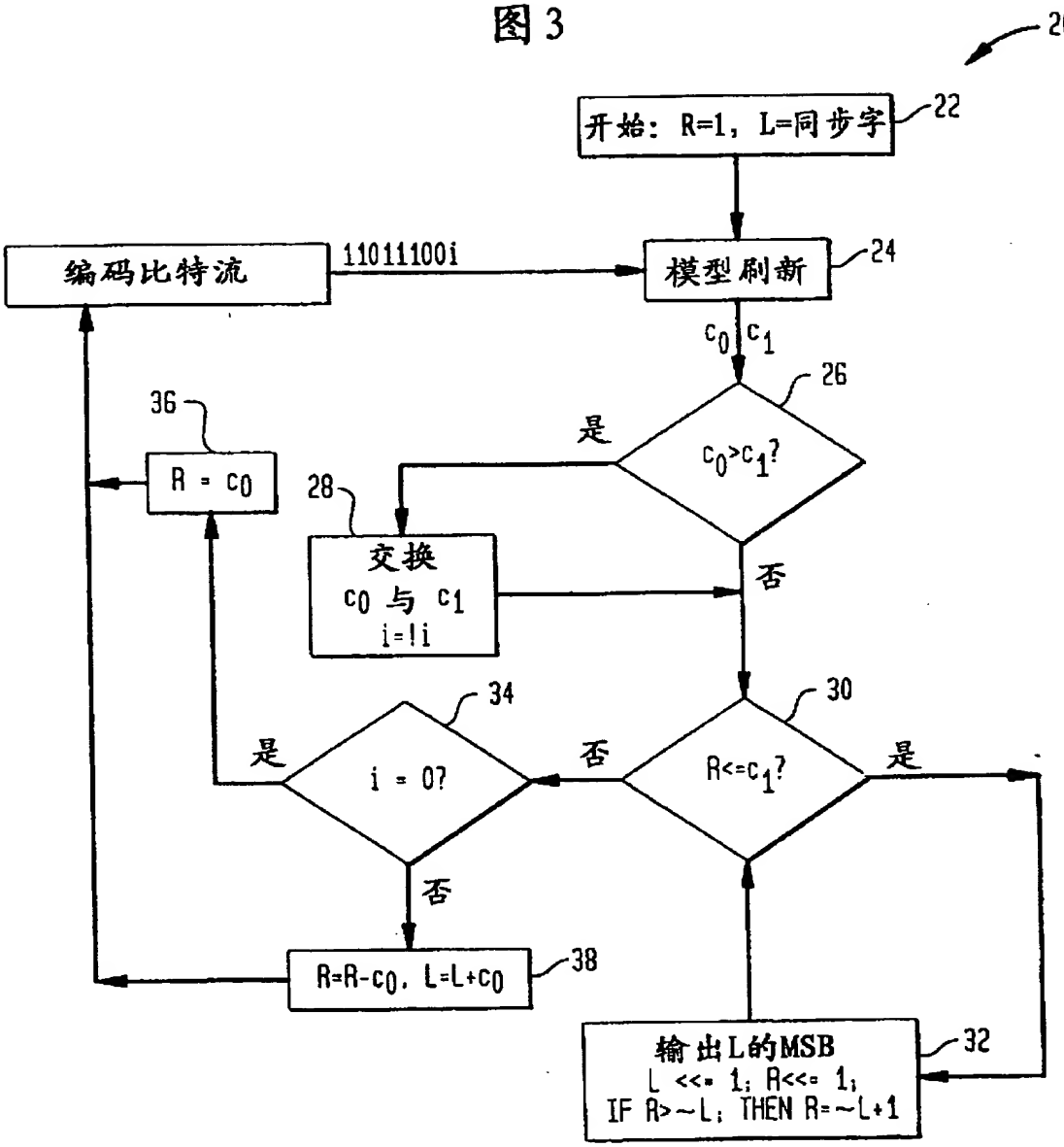

20
开始：R=1，L=同步字
22
编码比特流
11011100i
模型刷新
24
c_0 c_1
26
c_0>c_1?
是
否
36
R = c_0
28
交换
c_0 与 c_1
i=!i
30
R<=c_1?
是
否
34
i = 0?
是
否
R=R-c_0, L=L+c_0
38
输出L的MSB
L <<= 1; R<<= 1;
IF R>~L; THEN R=~L+1
32

图4

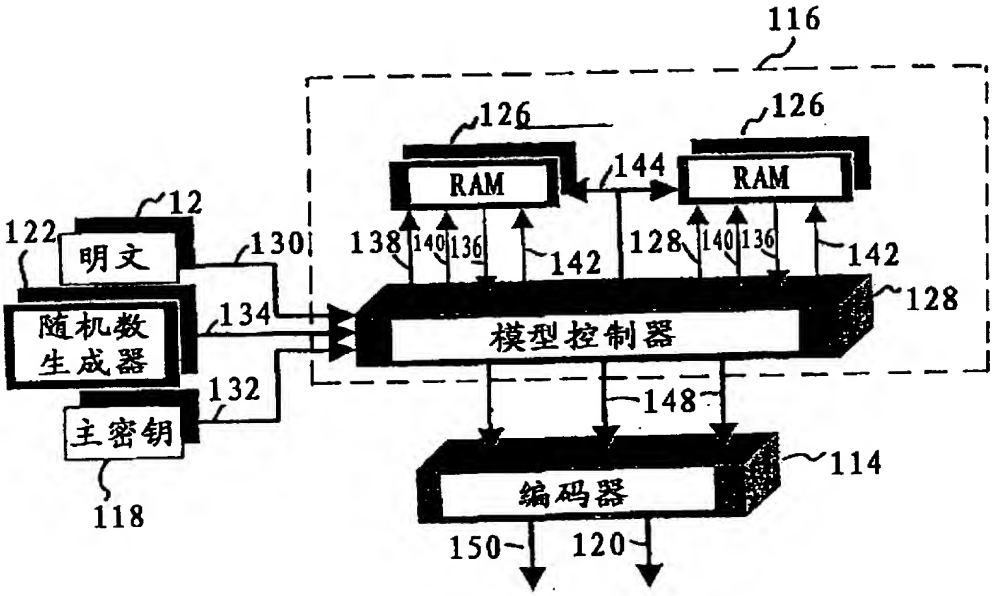

116
126
126
RAM
144
RAM
12
122
明文
130
138
140
136
142
128
140
136
142
128
随机数
生成器
134
模型控制器
132
148
主密钥
114
编码器
118
150
120

图5

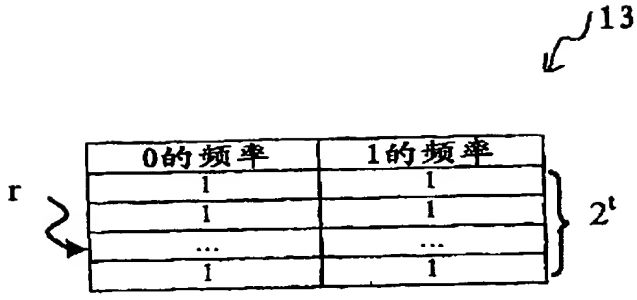

130
0的频率
1的频率
r
1
1
1
1
...
...
1
1
2^t

图6

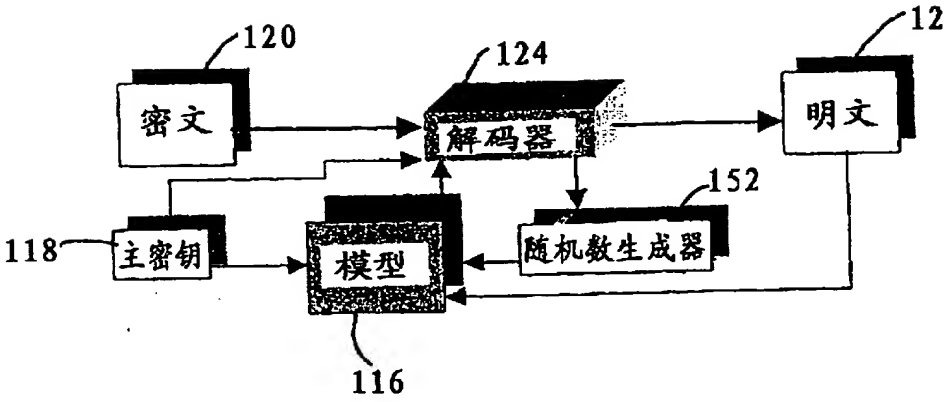

120
124
12
密文
解码器
明文
152
118
主密钥
模型
随机数生成器
116

图8

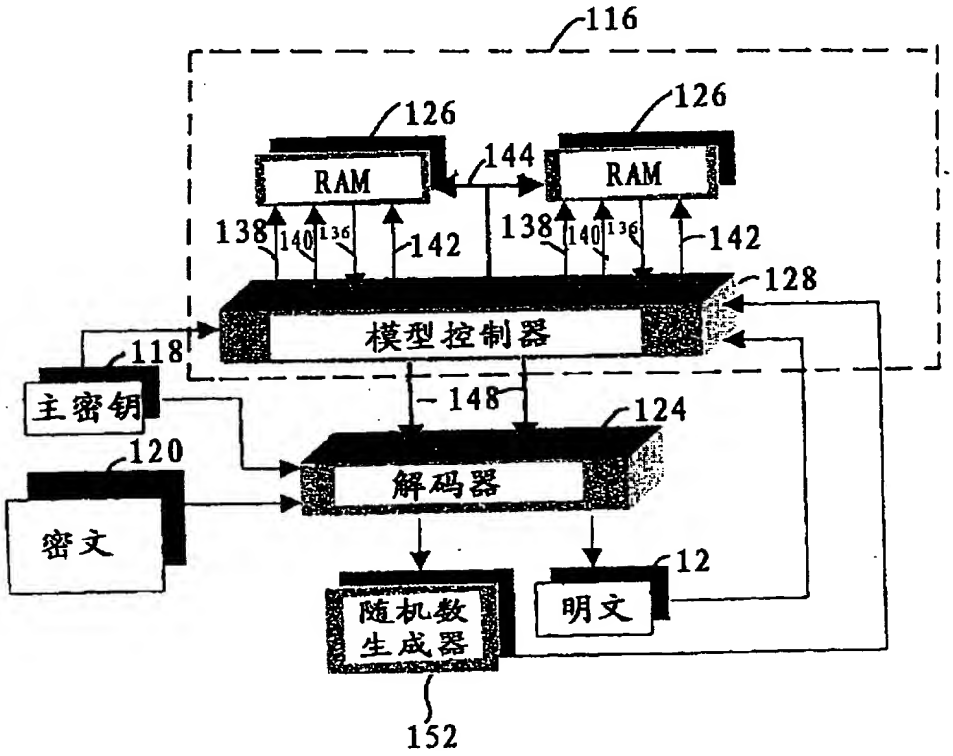
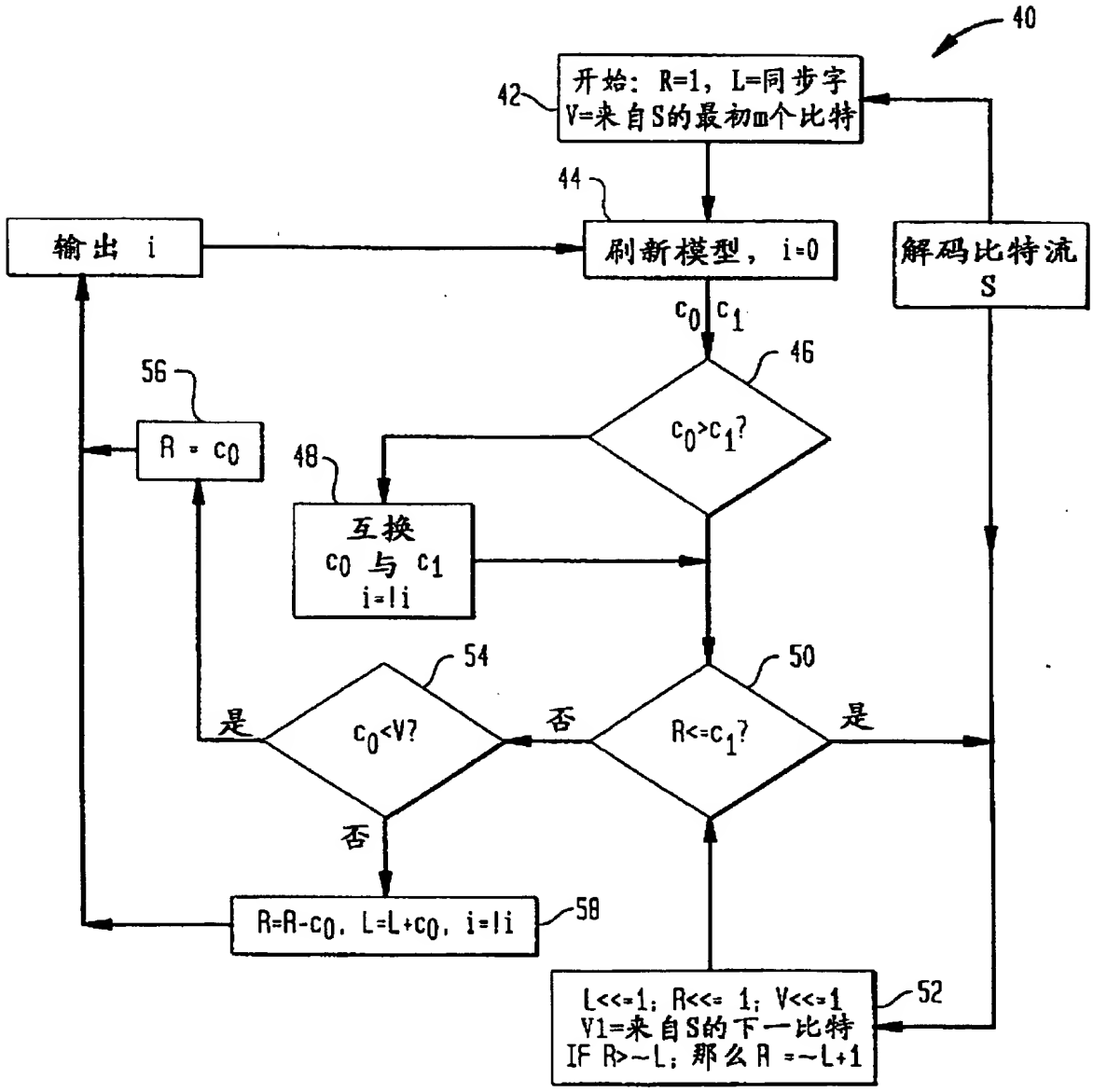

116
126
126
RAM
144
RAM
138
140
136
142
138
140
136
142
128
模型控制器
118
主密钥
148
124
120
解码器
密文
12
明文
随机数生成器
152

图7


40
开始：R=1，L=同步字
V=来自S的最初m个比特
42
44
刷新模型，i=0
输出 i
解码比特流
S
c_0 c_1
46
$c_0>c_1$?
56
R = c_0
48
互换
c_0 与 c_1
i=!i
54
$c_0<V$?
是
否
50
$R<=c_1$?
是
否
R=R-c_0, L=L+c_0, i=!i
58
L<<=1; R<<= 1; V<<=1
V1=来自S的下一比特
IF R>=L; 那么 R =-L+1
52